



Hardware Support for Dynamic Languages

Schleuniger, Pascal; Karlsson, Sven; Probst, Christian W.

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):
Schleuniger, P., Karlsson, S., & Probst, C. W. (2011). *Hardware Support for Dynamic Languages*. Poster session presented at 7th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems, Fiuggi, Italy. <http://www.hipeac.net/summerschool/>

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Motivation

- Dynamic programming languages:
 - enjoy increasing popularity
 - run on a virtual machine
 - have a long execution time
- Exploiting parallelism is difficult:
 - runtime execution, just-in-time compilation
 - no time for intensive code analysis
 - e.g. JavaScript is single threaded by design
- Software speculation is an effective method to exploit parallelism and speedup the code execution time
- We aim for hardware support for software speculation

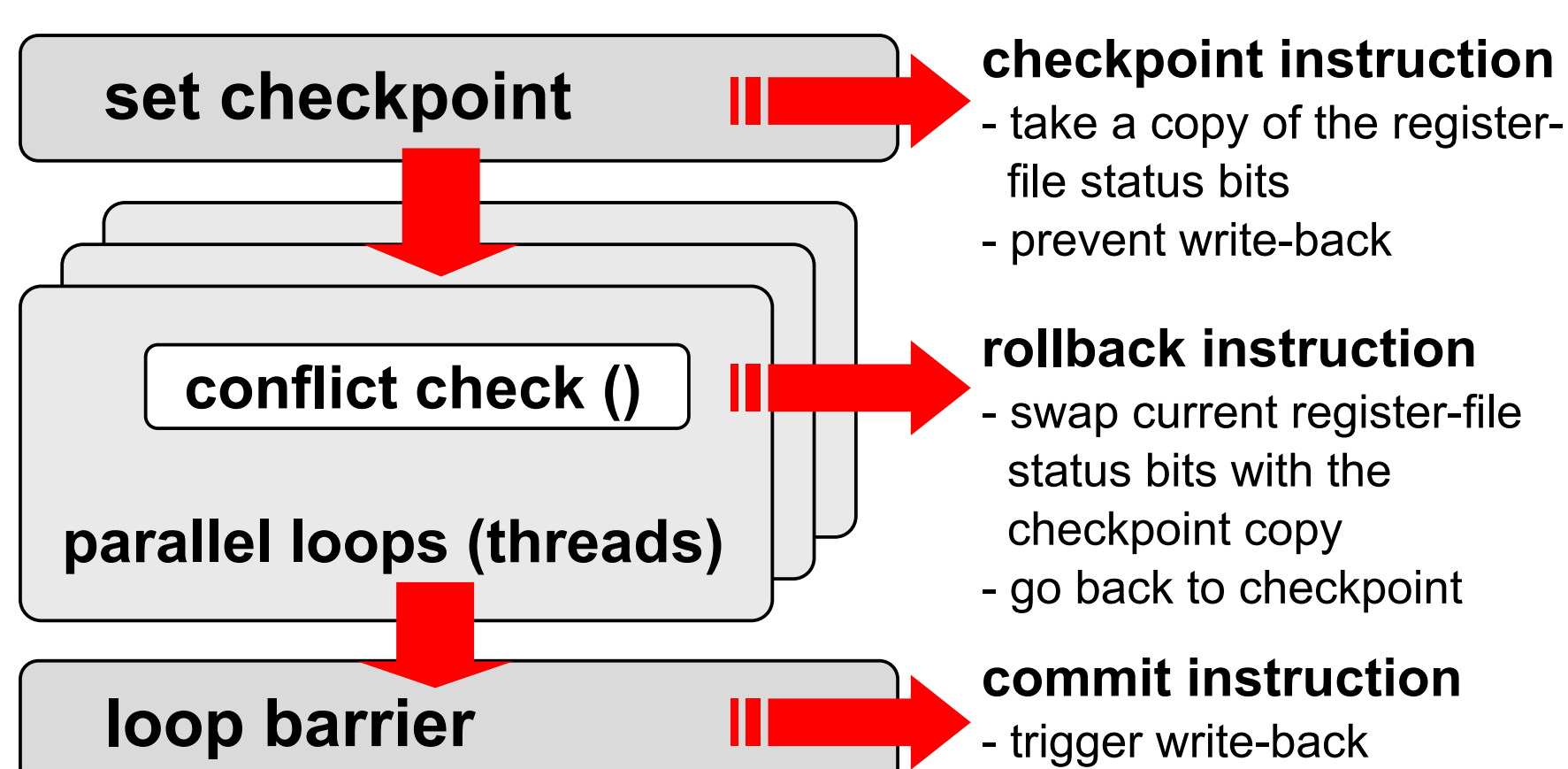
Predicated Instructions

- Instruction that is executed if a condition that is specified in the operation code is true, otherwise the instruction is annulled
- Predicated instruction example: convert a control dependence into data dependence


```
// C-code sequence:
if (a == 0){b = c + d ;}
// Predicated Instruction:
ADD b, c, d #a
```
- Eliminate some control dependencies
- Eases code analysis for parallelization process

Hardware Support for Rollback/Commit

- Software speculation can be applied for:
 - thread level, functions, types
- We aim for HW-support for rollback/commit:
 - shadow register-file with status bits
 - checkpoint/rollback/commit instructions
- Thread level speculation example: Loop iterations are handled as threads and are executed speculatively in parallel. If dependencies among threads are detected, the execution is rolled back to the checkpoint and executed sequentially instead.



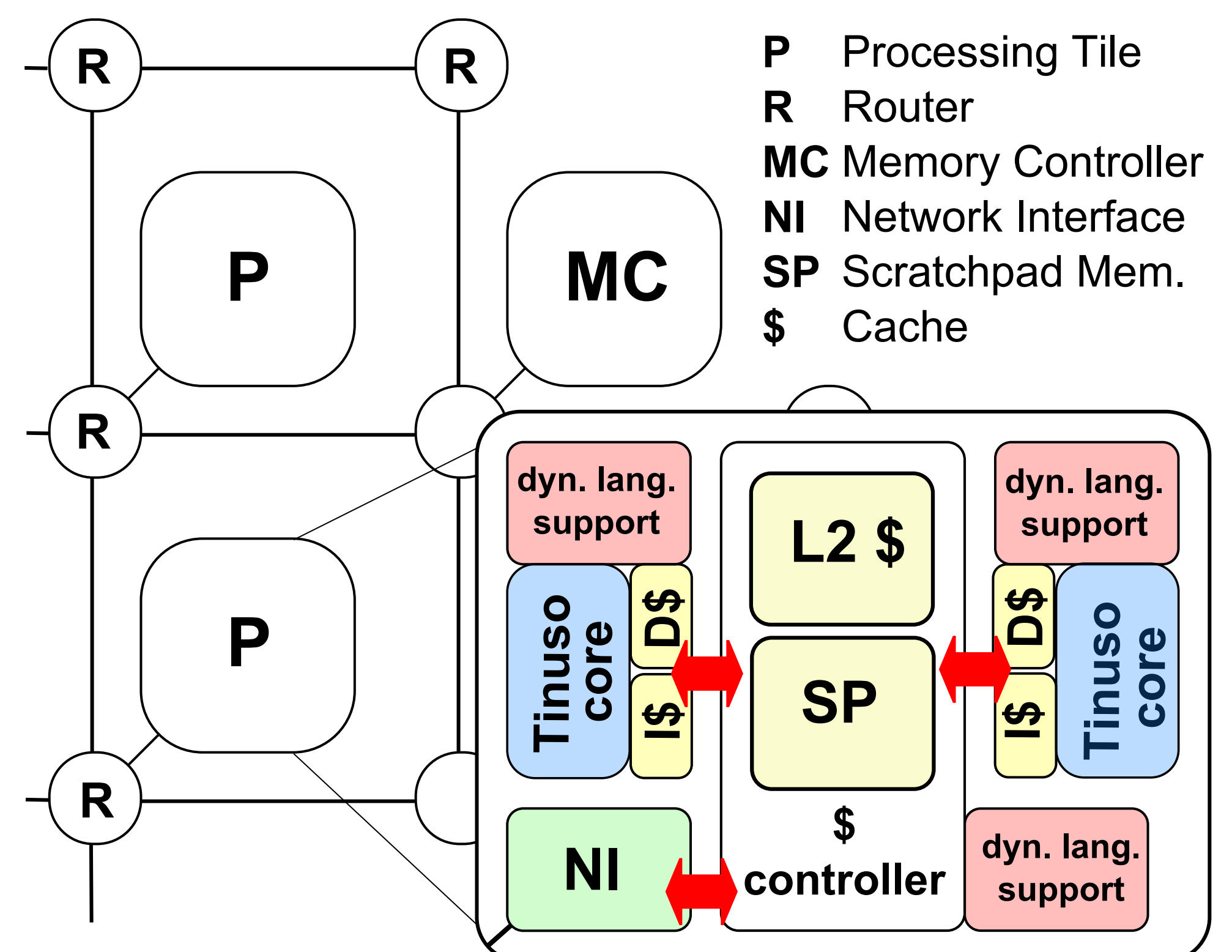
Hardware Support for Exceptions

- Suppress exceptions while code is executed speculatively
- Hardware support for conflict check when executing code speculatively (monitor data dependencies)

Hardware Support for Data Pre-fetching

- Speculative fetching of data and pre-computing
- Hides some of the memory access latency
- E.g. makes subsequent page loads of web applications faster

Hardware Experimentation Platform



- Tinuso Processor Core:
 - 32-bit, single-issue, RISC processor
 - 8-stage pipeline, full forwarding
 - predicated instructions
 - instruction- and datacache
 - barrel-shifter, multiplication unit
 - optimized for FPGA implementation
 - Xilinx Virtex6(-3): 370MHz
 - Processing Tile:
 - two Tinuso cores in one processing tile
 - network-interface
 - 2-nd level cache*
 - scratchpad memory*
 - hardware support for cache coherency*
 - Network-on-Chip:
 - packet-switched, mesh-4 network
 - non-blocking, XY-routing
- *implementation in progress